Towards a Zookeeper-less Pulsar

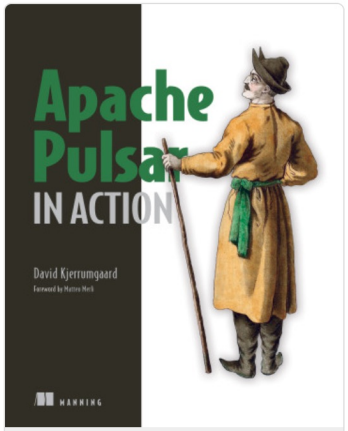
David Kjerrumgaard| Developer Advocate

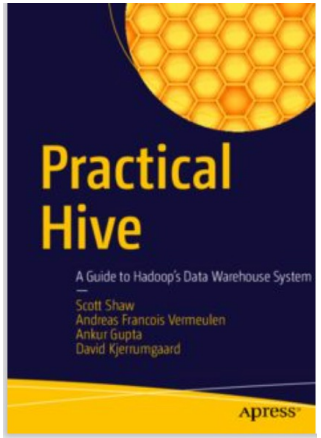
David Kjerrumgaard
Developer Advocate

- Apache Pulsar Committer
- Former Principal Software Engineer on Splunk's messaging team that is responsible for Splunk's internal Pulsar-as-a-Service platform.
- Former Director of Solution Architecture at Streamlio.
- Global practice director of Professional Services at Hortonworks.

David Kjerrumgaard
Author

- Author of **Pulsar In Action.**
- Co-Author of ***Practical Hive***

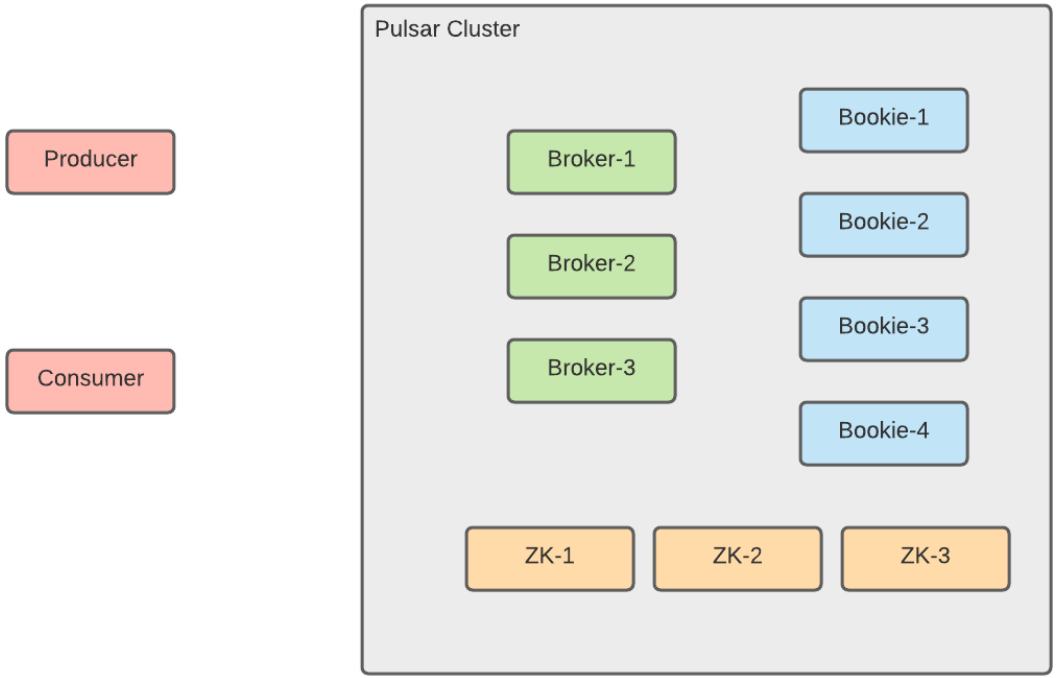




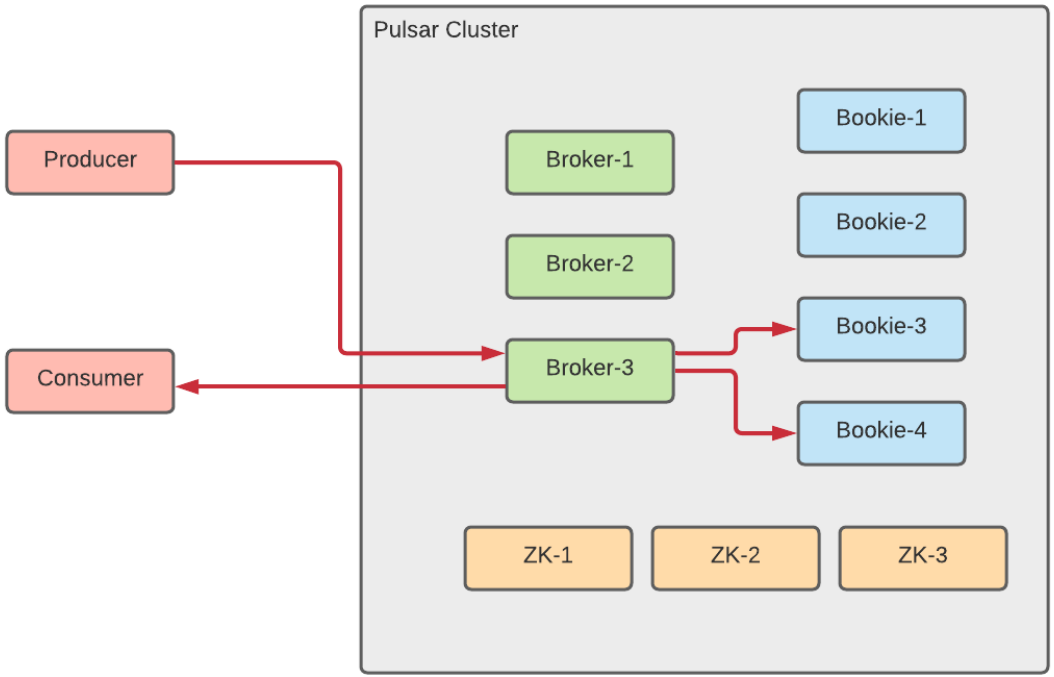
https://streamnative.io/download/manning-ebook-apache-pulsar-in-action

Pulsar and Metadata

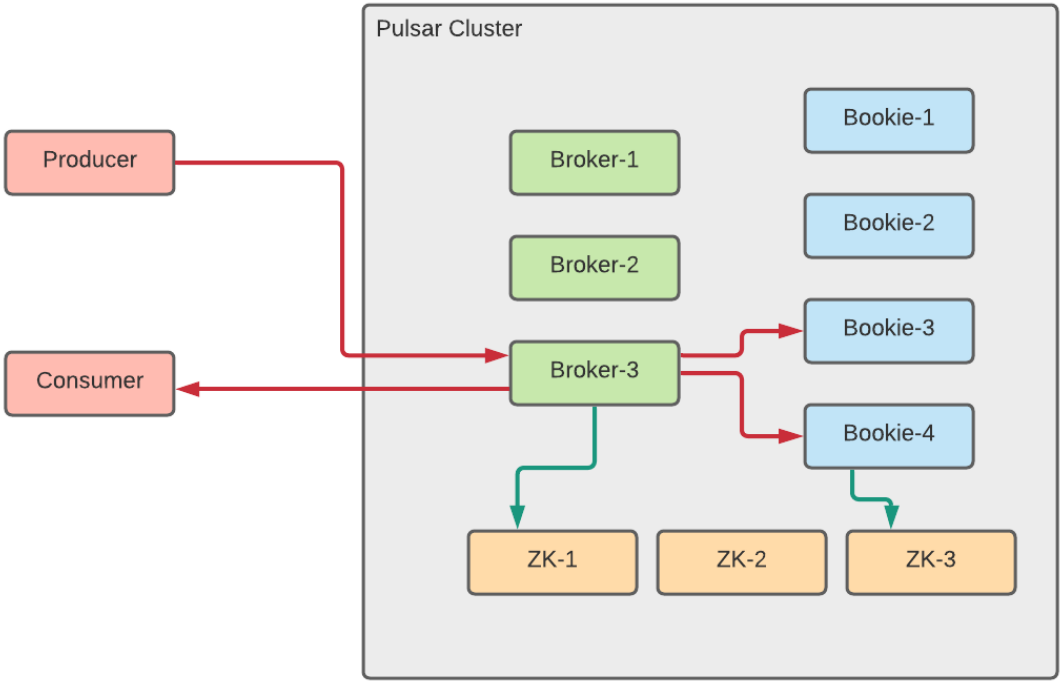
Pulsar Cluster Overview



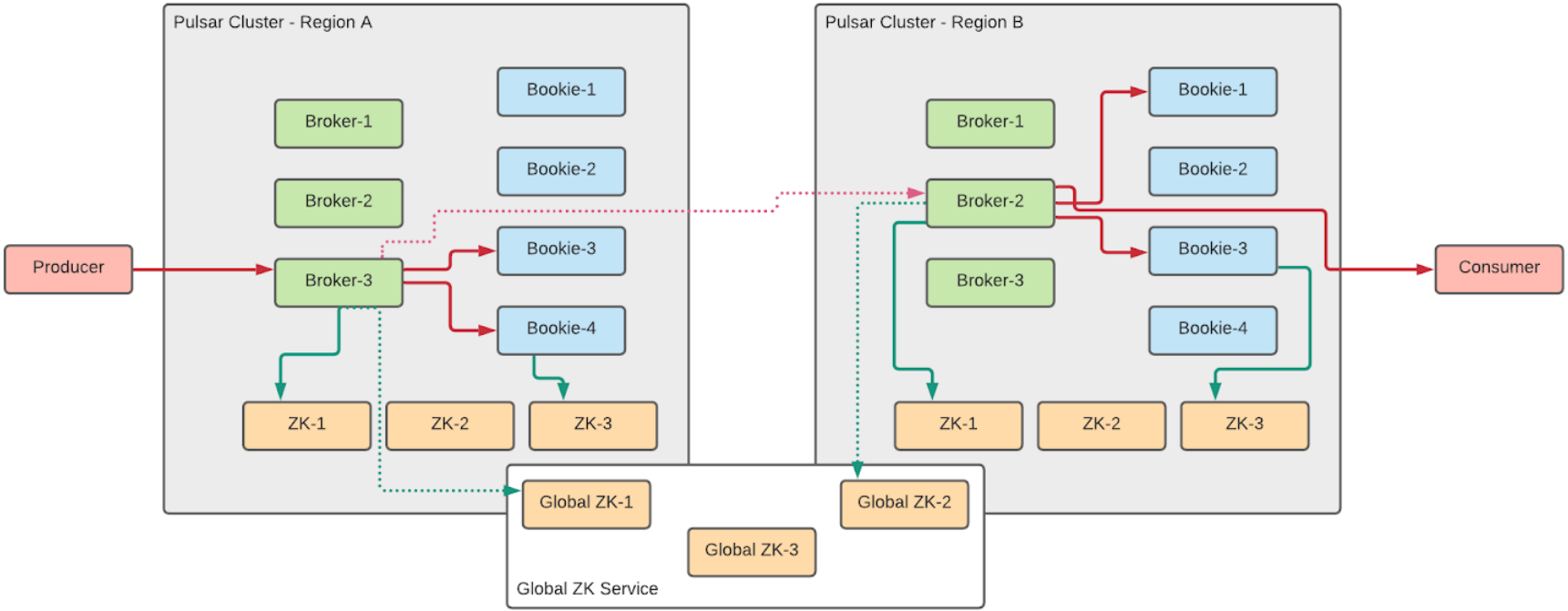
The Data Path



The Metadata Path



Geo-Replication



Pulsar’s Metadata

Categories of Metadata

Pointers To Data

Managed Ledgers

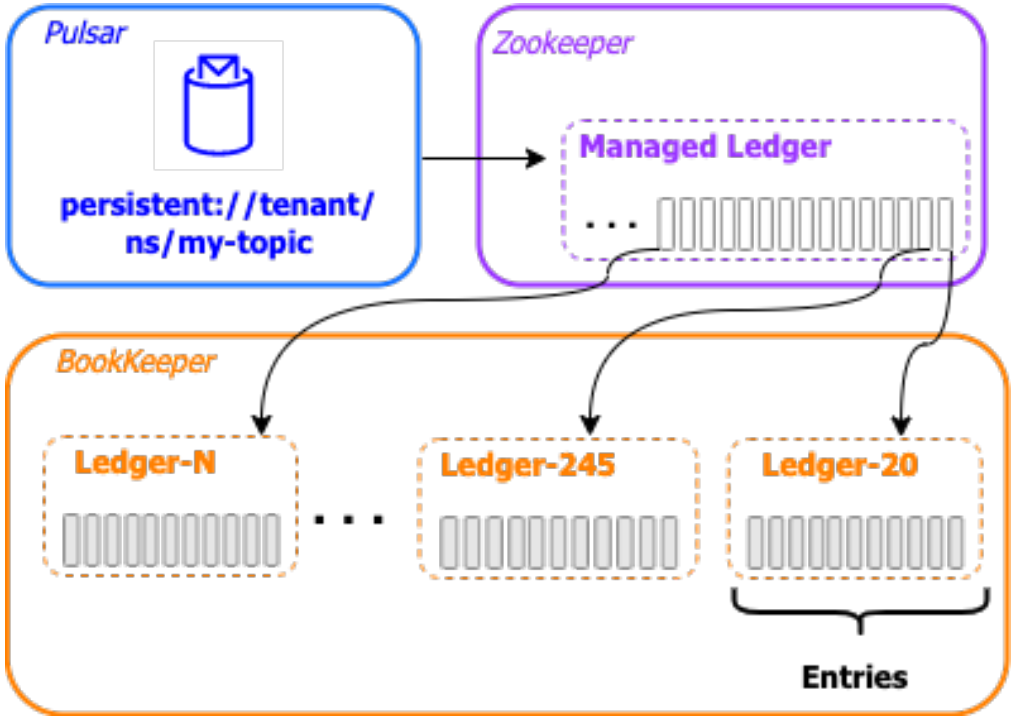
- Each persistent topic is associated with an ordered list of ledgers known as a *managed ledger*.

Ledger Metadata

- Each BookKeeper ledger has associated metadata that tracks the state of the ledger, and which bookies have a replica.

Managed Ledger

- An append-only list of ledger IDs that hold the topic's data.
- Only updated when a segment rolls-over, e.g., once every 50k entries or 4 hours.



The Managed Ledger

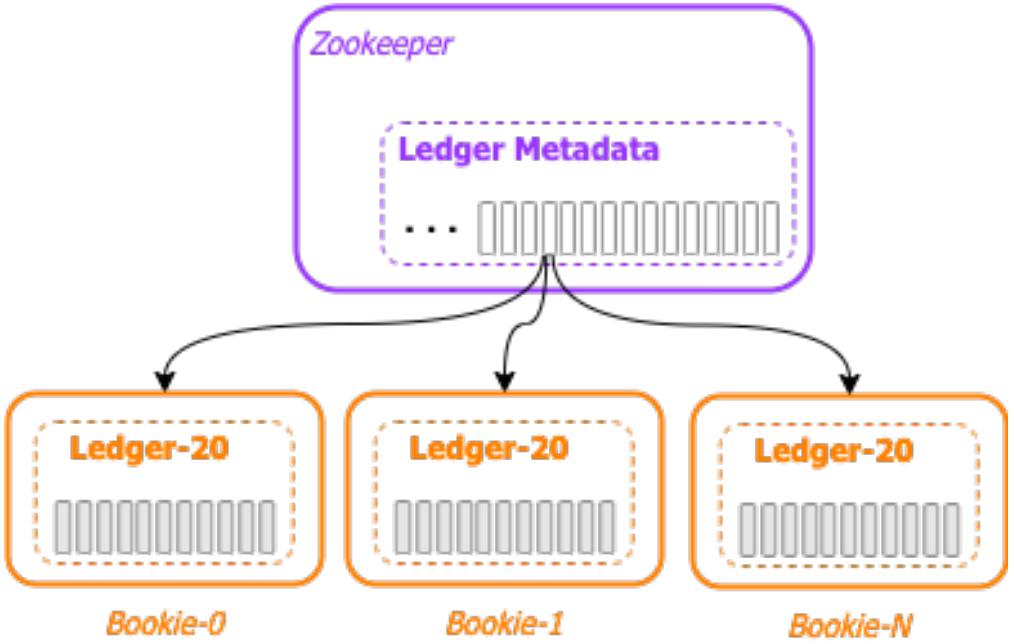
- Stored inside ZK in a hierarchical manner e.g.,
  - `/managed-ledgers/<tenant>/<ns>/persistent/<topic>`
- Administered via `./bin/pulsar-managed-ledger-admin`
- Examine the data using `./bin/pulsar-admin topics info-internal $TOPIC`

Managed Ledger Example


```
{ "ledgers" :
  [{"ledgerId":1234,"entries":1000,"size":433111,"offloaded":false},
   {"ledgerId":5579,"entries":50000,"size":9433111,"offloaded":false}
   . . .
  ],
 "schemaLedgers":[],
 "compactedLedger":
    {"ledgerId":-1,"entries":-1,"size":-1,"offloaded": false}
}
```

Ledger Metadata

- Tracks the state of the ledger, and which bookies have a replica.
- New entry added only when a segment rolls-over, e.g., once every 50k entries or 4 hours.



Ledger Metadata

- Stored inside ZK in a hierarchical manner e.g.,
  - `/ledgers/00`
- Administered via `./bin/bookkeeper shell`
- Examine the data using `./bin/bookkeeper shell ledgermetadata -ledgerid <LEDGER-ID>`

Ledger Metadata Example

```
LedgerMetadata{ formatVersion=3, ensembleSize=2,
  writeQuorumSize=2, ackQuorumSize=2, state=CLOSED,
  length=1738964, lastEntryId=1611,
  digestType=CRC32C, password=base64:,
  ensembles={
    0=[bookie-1:3181, bookie-2:3181],
    1000=[bookie-5:3181, bookie-2:3181]
 },
 customMetadata={
   component=base64:bWFuYWdlZC1sZWRnZXI=,
   pulsar/managed-ledger=base64:cHVibGlR=,
   application=base64:cHVsc2Fy
  }
}
```

Service Discovery

Bookies

- Find available bookies.
- Which bookies are in read-only mode?

Brokers

- Find available brokers
- Discover which broker owns a particular topic
- What is the current load on each broker?

Available Brokers

```
ls /loadbalance/brokers

[pulsar-full-broker-0.pulsar-full-
broker.pulsar.svc.cluster.local:8080, pulsar-full-broker-
1.pulsar-full-broker.pulsar.svc.cluster.local:8080]
```

Broker Assignment

```
get /namespace/public/default/0x80000000_0x90000000

{"nativeUrl":"pulsar://pulsar-full-broker-1.pulsar-full-broker.pulsar.svc.cluster.local:6650","httpUrl":"http://pulsar-full-broker-1.pulsar-full-broker.pulsar.svc.cluster.local:8080","disabled":false,"advertisedListeners":{}}
```

Current Broker Load

```
get /loadbalance/broker-time-average/pulsar-full-broker-0.pulsar-full-broker.pulsar.svc.cluster.local:8080

{"shortTermMsgThroughputIn":0.0,"shortTermMsgThroughputOut":0.0,"shortTermMsgRateIn":0.0,"shortTermMsgRateOut":0.0,"longTermMsgThroughputIn":0.0,"longTermMsgThroughputOut":0.0,"longTermMsgRateIn":0.0,"longTermMsgRateOut":0.0}
```

Current Bundle Load

```
get /loadbalance/bundle-data/public/default/0x20000000_0x30000000

{"shortTermData":{"maxSamples":10,"numSamples":10,"msgThroughputIn"
:62.25184125922071,"msgThroughputOut":7.60488525462946 5E-
11,"msgRateIn":0.0577580340194091 4,"msgRateOut":1.768577966192899E-
12},"longTermData":{"maxSamples":1000,"numSamples":234,"msgThroughp
utIn":920.1840888229265,"msgThroughputOut":0.0577503597307179 25,"ms
gRateIn":0.8537582637120998,"msgRateOut":0.0013430316216446019},"to
pics":1}
```

System Configuration

- Allow for dynamic settings
- Features can be activated/deactivated without restarting brokers
- Keep isolation information
- Maintain tracking of (bookie -> rack) mapping

System Policies

```
get /admin/policies/pulsar/system

{"auth_policies":{"namespace_auth":{},"destination_auth":{},"subscription_auth_roles":{}}
,"replication_clusters":["pulsar-
full"],"bundles":{"boundaries":["0x00000000","0x10000000","0x20000000","0x30000000","0x40
000000","0x50000000","0x60000000","0x70000000","0x80000000","0x90000000","0xa0000000","0x
b0000000","0xc0000000","0xd0000000","0xe0000000","0xf0000000","0xffffffff"],"numBundles":
16},"backlog_quota_map":{},"clusterDispatchRate":{},"topicDispatchRate":{},"subscriptionD
ispatchRate":{},"replicatorDispatchRate":{},"clusterSubscribeRate":{},"publishMaxMessageR
ate":{},"latency_stats_sample_rate":{},"deleted":false,"encryption_required":false,"subsc
ription_auth_mode":"None","offload_threshold":-
1,"schema_compatibility_strategy":"UNDEFINED","schema_validation_enforced":false,"subscri
ption_types_enabled":[],"properties":{}}
```

Provisioning Configuration

- Metadata for Tenants, Namespaces
- Policies to apply to namespaces
- Authorization definitions
- Highly-Cacheable metadata

Namespace Policies

```
get /admin/policies/public/default

{"auth_policies":{"namespace_auth":{},"destination_auth":{},"subscription_auth_roles":{}}
,"replication_clusters":["pulsar-
full"],"bundles":{"boundaries":["0x00000000","0x10000000","0x20000000","0x30000000","0x40
000000","0x50000000","0x60000000","0x70000000","0x80000000","0x90000000","0xa0000000","0x
b0000000","0xc0000000","0xd0000000","0xe0000000","0xf0000000","0xffffffff"],"numBundles":
16},"backlog_quota_map":{"destination_storage":{"limit":-1,"limitSize":-1,"limitTime":-
1,"policy":"producer_request_hold"}},"clusterDispatchRate":{},"topicDispatchRate":{},"sub
scriptionDispatchRate":{},"replicatorDispatchRate":{},"clusterSubscribeRate":{},"publishM
axMessageRate":{},"latency_stats_sample_rate":{},"deleted":false,"encryption_required":fa
lse,"subscription_auth_mode":"None","offload_threshold":-
1,"schema_compatibility_strategy":"UNDEFINED","schema_validation_enforced":false,"subscri
ption_types_enabled":[],"properties":{}}
```

Distributed Coordination

- Acquire a lock over a particular resource
  - Ownership of group of topics
  - Signaling that some work on a particular resource is in progress
    - BK auto-recovery
- Leader election
  - Establish a single leader designed to perform some tasks
    - Load manager designates a leader that
  - Failover to other available nodes

What's up with Zookeeper?

Apache Zookeeper

- Consensus based “database”; data is replicated consistently to a quorum of nodes.
- It is not horizontally scalable; increasing the ZK cluster size does ***not*** increase the write capacity!
- All data is kept in memory in every node - Not very GC friendly
- It takes periodic snapshots of the entire dataset.

Apache Zookeeper Issues

- The amount of metadata that can be stored in ZK is ~5GB
- Tuning and operating ZK to work with big datasets is not trivial.
- Requires deep knowledge of ZK internals.
- In cloud and containerized environments, leader election can sometime take few minutes due to:
  - Issues with DNS, software-defined-networking and sidecar TCP proxies.

Reasons to remove Zookeeper

- Big clusters → we don't want to have a hard limit of the amount of metadata
  - A horizontally scalable metadata store is more suited
- Small clusters → remove overhead of running ZK
  - Less components to deploy
  - Easier operations

PIP-45

A Multi-step Plan

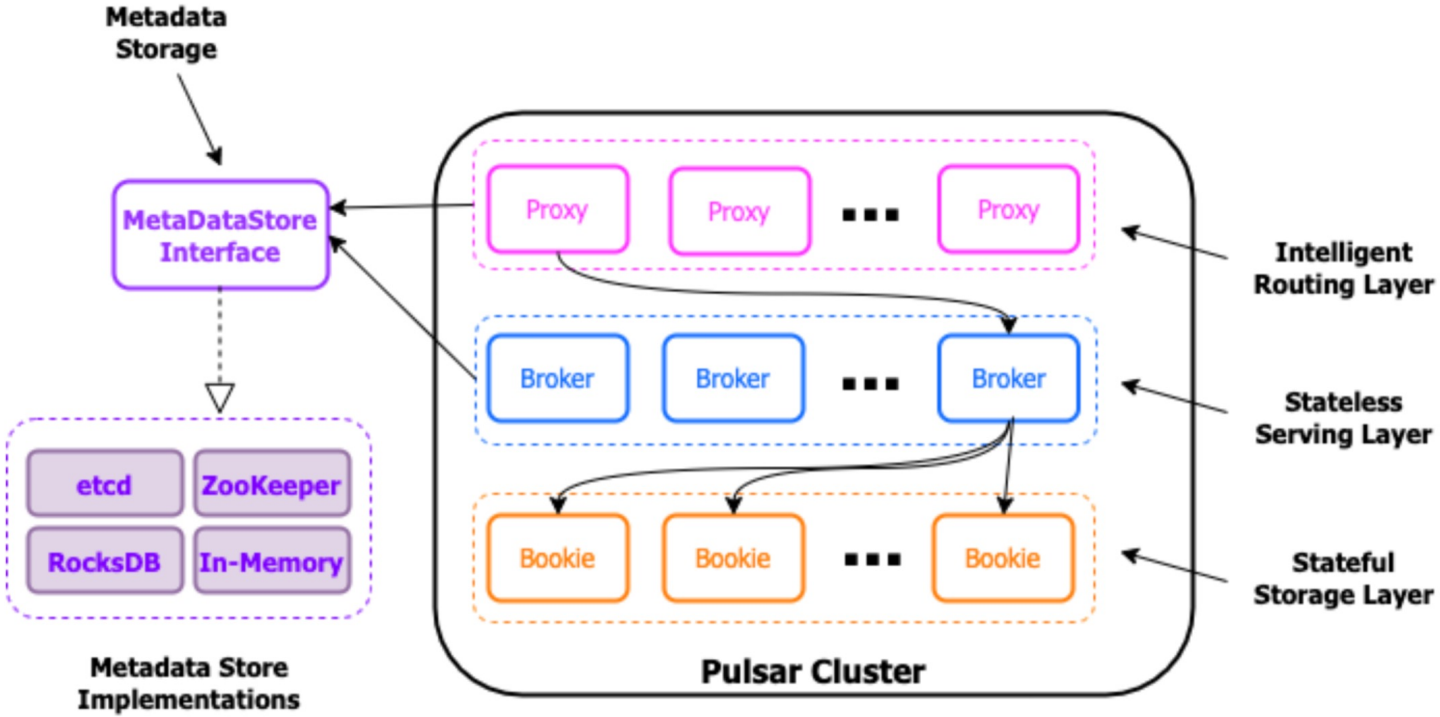
Design Decisions

- Let’s not implement directly in Pulsar brokers
- Let’s not rewrite Paxos/Raft again
- Assume the facilities of a cloud-native environment
- Design for auto-tuning, from tiny to huge without admin intervention

Pluggable Metadata Backend

- Instead of direct usage of ZooKeeper APIs, we have abstracted all the accesses through a single generic API.
- This API has multiple implementations:
  - ZooKeeper
  - Etcd
  - RocksDB (for standalone)
  - Memory (for unit tests)

Pluggable Metadata Backends



Metadata Semantics

- We have identified 2 main patterns of access to the metadata
  - Simple key-value access + notifications
  - Complex coordination

Key-Value Access

- MetadataStore → Key-value store access
  - put() – get() – delete()
  - Values are byte[]
  - Users can register for notifications
- MetadataCache → Object cache on top of MetadataStore

Coordination Services

- Contains primitives for “cluster coordination”
- High-level API that hides all the complexities
  - ResourceLock – Distributed lock over a shared resource
  - LeaderElection – Elect a leader among a set of peers
  - DistributedCounter – Generate unique IDs

Metadata Store Administration

Configuration

- The metadata store of each Pulsar instance should contain the following two components:
  - A local metadata store ensemble (`metadataStoreUrl`) that stores cluster-specific configuration and coordination, such as which brokers are responsible for which topics as well as ownership metadata, broker load reports, and BookKeeper ledger metadata.
  - A configuration store quorum (`configurationMetadataStoreUrl`) stores configuration for clusters, tenants, namespaces, topics, and other entities that need to be globally consistent.

ZooKeeper as the Metadata Store

- Pulsar metadata store can be deployed on a separate ZooKeeper cluster or deployed on an existing ZooKeeper cluster.

- Add the following parameters to the conf/broker.conf or conf/standalone.conf file.

```
metadataStoreUrl=zk:my-zk-1:2181,my-zk-2:2181,my-zk-3:2181

configurationMetadataStoreUrl=zk:my-global-zk-1:2181,my-
global-zk-2:2181,my-global-zk-3:2181
```

Etcd as the Metadata Store

- Pulsar metadata store can be deployed on an existing Etcd cluster.
- Add the following parameters to the conf/broker.conf or conf/standalone.conf file.

```
metadataStoreUrl=etcd:http://my-etcd-1:2379,http://my-etcd-2:2379,http://my-etcd-3:2379

configurationMetadataStoreUrl=etcd:my-global-etcd-1:2379,my-global-etcd-2:2379,my-global-etcd-3:2379

# metadataStoreConfigPath=/path/to/file
```

Etcd – cont.

- The metadataStoreConfigPath parameter is required when you want to use the following advanced configurations.

```
useTls=false
tlsProvider=JDK
tlsTrustCertsFilePath=
tlsKeyFilePath=
tlsCertificateFilePath=
authority=
```

RocksDB as the Metadata Store

- Pulsar metadata store can be deployed on a new or existing RocksDB database.

- Add the following parameters to the conf/broker.conf or conf/standalone.conf file.

```
metadataStoreUrl=rocksdb://data/metadata
# metadataStoreConfigPath=/path/to/file
```

RockDB – cont.

- The metadataStoreConfigPath parameter is required when you want to use advanced configurations.

```
[DBOptions]
  stats_dump_period_sec=600
  max_manifest_file_size=18446744073709551615
  bytes_per_sync=8388608
  delayed_write_rate=2097152
  WAL_ttl_seconds=0
  WAL_size_limit_MB=0
  max_subcompactions=1
```

In-Memory Metadata Store

- Pulsar metadata store can be be hosted in local memory for things like unit testing, etc.

- Add the following parameters to the conf/broker.conf or conf/standalone.conf file.

```
metadataStoreUrl=memory://local
```

Successes Enabled In Pulsar 2.10

Metadata Session Revalidation

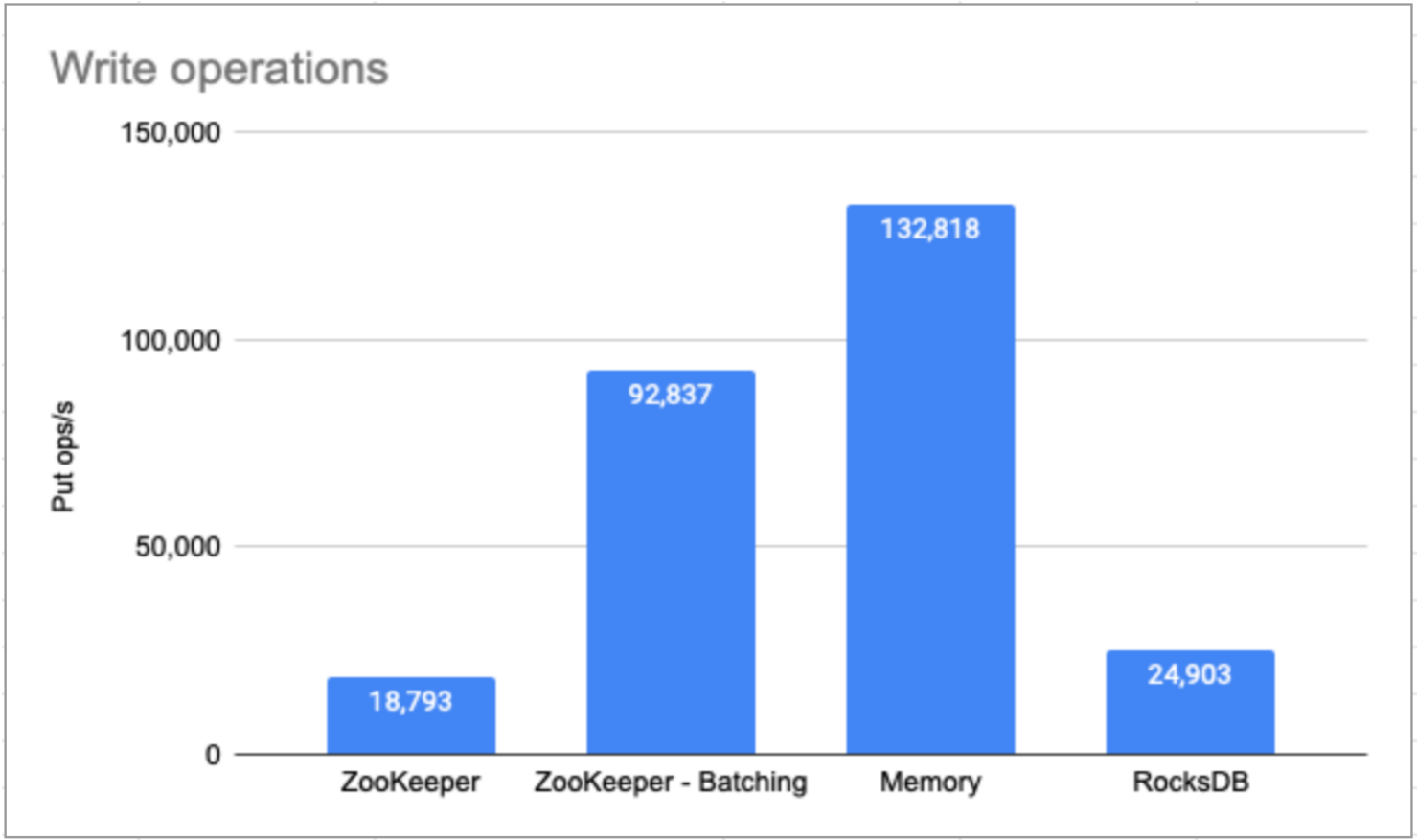
When we lose a ZooKeeper session (or similarly an Etcd lease), we can re-validate it later, without having to restart Pulsar brokers.

This is a major cluster stability improvement.

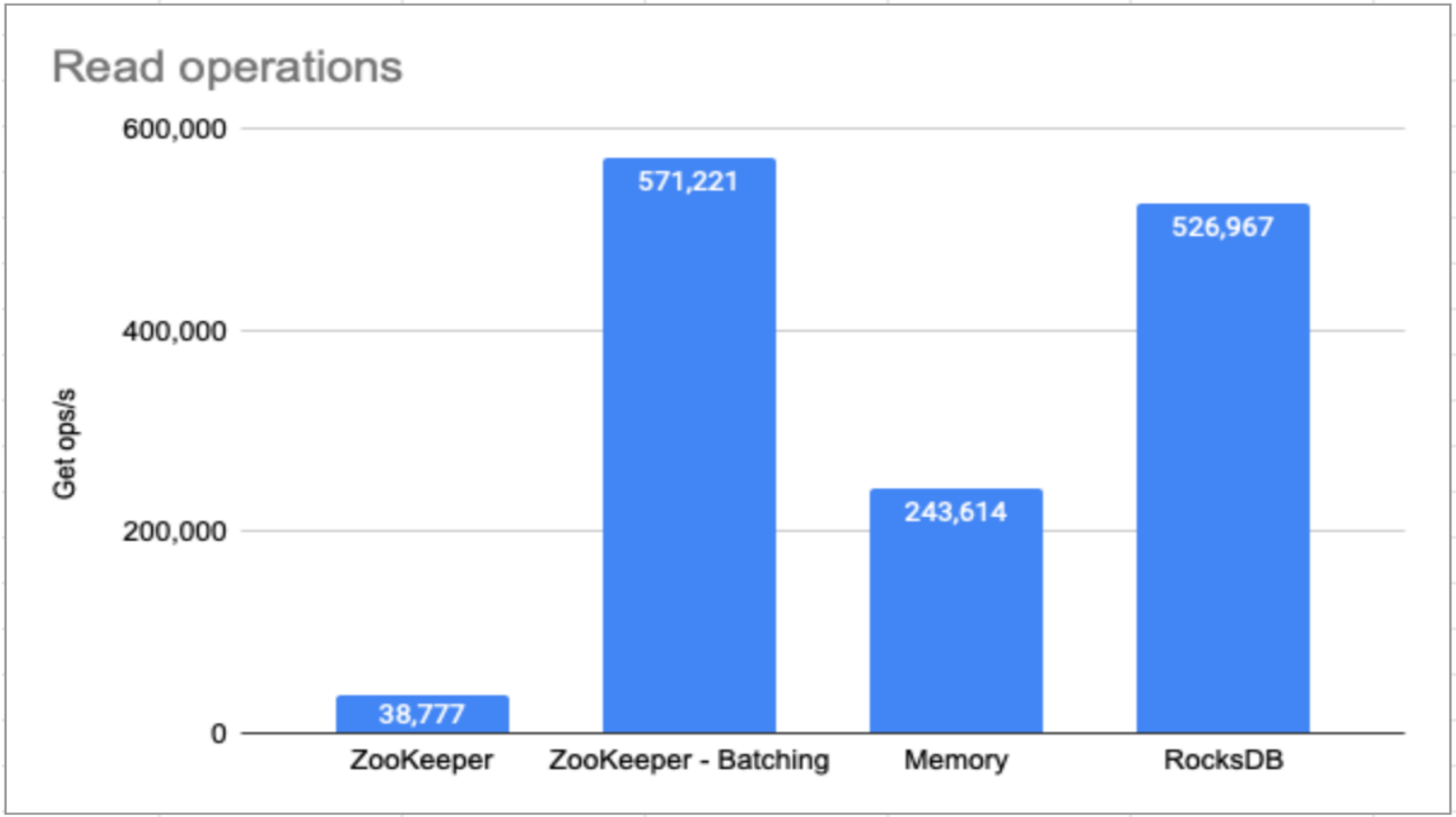
Batching of Metadata

- All the metadata read and write operations are happening through a single access point
- Accumulate operations into a queue and use underlying API for bulk access (e.g.: ZK “multi” or Etcd transactions)
- This is a major improvement in metadata throughput

Increased Metadata Writes



Increased Metadata Reads



Configuring Batching

- To enable batch operations on the metadata store, you can configure the following parameters in the conf/broker.conf or conf/standalone.conf file.

```
# Whether we should enable metadata operations batching
metadataStoreBatchingEnabled=true

# Maximum delay to impose on batching grouping
metadataStoreBatchingMaxDelayMillis=5

# Maximum number of operations to include in a singular batch
metadataStoreBatchingMaxOperations=1000

# Maximum size of a batch
metadataStoreBatchingMaxSizeKb=128
```

What’s Next?

Goals for the Metadata Service

- Transparent horizontal scalability
- Ease of operations (add/remove nodes)
- No need for global linearizable history
- Scale up to 100 GB of total data set
- Read - Write rates scalable to ~1M ops/s
- Latency target: reads 99pct < 5ms — writes 99pct < 20ms

Expected Results

- The ultimate goal is to achieve a 10x increase in number of topics in a cluster
- A small Pulsar cluster should be able to support millions of topics
- Handling of metadata is the biggest obstacle
- It’s not the only factor though. We are also working on metrics, lookups, overhead of single topic and global memory limits

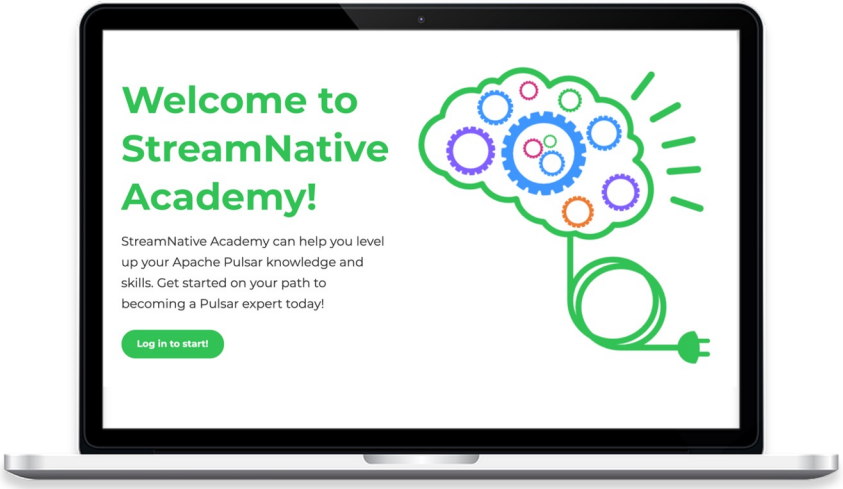
Thanks for attending!

Scan the QR Code to learn more about Apache Pulsar.

Now Available

On-Demand Pulsar Training

Academy.StreamNative.io



We're Hiring

streamnative.io/careers/

Questions

Let’s Keep in Touch!

David Kjerrumgaard

Developer Advocate

@Dkjerrumg1

https://www.linkedin.com/davidkj

https://github.com/david-streamlio

StreamNative